APPARATUS AND METHOD FOR RENDERING CHARACTERS INTO A MEMORY

Ins A1

FIELD OF THE INVENTION

The present invention relates to generally to a computer system and more particularly to rendering font characters into memory in a computer system in an efficient manner.

BACKGROUND OF THE INVENTION

The constant emphasis in the design of computer systems has been on making such systems faster and able to handle larger amounts of information so that they may accomplish more work. The ability of computers to do more work is also enhanced by their ability to do more different kinds of work. For example, computers have become of much more use in many more activities with the advent of computer graphics displays which allow the presentation of not only numbers and text materials but pictures and graphs representing the meanings of those numbers and adding tot he meaning of the text material. It has gradually become the belief of a great number of people in the computer industry that a graphical output should be presented on almost all computer systems.

In a typical computer system, a graphics controller sends video signals to monitors and also controls a frame buffer memory system. Fonts representing text, for example, are written into a frame buffer associated with the computer system. The graphics controller typically extracts the characters of the font to scan them onto a display or monitor of the computer system. A first system for rendering font characters into the frame buffer on system memory requires that the font characters be downloaded directly from a central processing unit (CPU) via the graphics controller. Downloading the characters directly from the CPU can be a very slow process because

an image of a character typically requires many bytes per pixel.

In a second embodiment, a glyph is provided. A glyph is a compressed representation of the font for each byte of a font. Typically, in a glyph a bit represents a pixel of information. Accordingly, each pixel represented on the frame buffer is represented by one bit of information provided by the CPU, rather than a byte or bytes of information being rendered in to the memory as required in the first above-mentioned system. Accordingly, it is possible in accordance with the second embodiment to provide for lossless compression and therefore a faster render rate is possible.

In such a system, the graphics controller will select between two color registers, typically referred to as a foreground register and a background register. The graphics controller receives the bits from the CPU representative of the particular character of a particular font and based upon the whether a bit represents foreground or background color, the appropriate register renders the byte or bytes of information for that pixel. The lossless compression is possible because font characters are typically one color (for example, black) and therefore only two different colors need to be provided (i.e., black for foreground and white for background). Although the second embodiment has some advantages over the first above described embodiment, it still has some significant disadvantages which will be described below.

A system in accordance with the second embodiment, requires that the CPU transfer several pieces of information to graphics controller for every font character that is to be rendered into memory. Accordingly, the CPU must find the fonts, find the character in the font, and then figure out how many bytes to transfer and transfer this information across the bus. In so doing, several different transfers are required to render a group of font characters into memory. Therefore, the bandwidth required for the transfers becomes a significant issue.

Accordingly, what is needed is a system that allows for increased throughput of fonts in a computer system to allow more efficient rendering of font characters thereof. The system must be efficient, easily implemented and a cost effective alternative to existing systems.

The present invention addresses such a need.

SUMMARY OF THE INVENTION

A system and method for rendering fonts into a memory is disclosed. The system and method comprises a data structure located within the memory. The data structure includes at least one font array. The method and system includes a graphics controller for accessing at least one font array in the memory and for rendering characters of at least one font array into the appropriate locations in the memory to be scanned onto a monitor.

Accordingly, a system and method in accordance with the present invention provides for a plurality of font arrays to be provided within a memory of a computer system. The memory could be the frame buffer, system memory or any other memory within the computer system. The graphics controller includes a mechanism which allows for a font array to be accessed by the graphics controller. The graphics controller also includes a mechanism for allowing each font character to be rendered into the memory. In so doing, the number of transfers from the CPU is significantly reduced.

BRIEF DESCRIPTION OF THE DRAWINGS

Figure 1 is a block diagram of a conventional computer system.

Figure 2 illustrates rendering of pixels of information into a frame buffer directly by a central processing unit (CPU).

Figure 3 illustrates a compressed font character produced in accordance with a second conventional technique.

Figure 4 illustrates a graphics controller and a frame buffer which operates in accordance with the second conventional technique.

Figure 5 illustrates a graphics controller and a frame buffer in accordance with the present invention.

DETAILED DESCRIPTION

The present invention relates to generally to a computer system and more particularly to rendering font characters into memory in a computer system in an efficient manner. The following description is presented to enable one of ordinary skill in the art to make and use the invention and is provided in the context of a patent application and its requirements. Various modifications to the preferred embodiment and the generic principles and features described herein will be readily apparent to those skilled in the art. Thus, the present invention is not intended to be limited to the embodiment shown but is to be accorded the widest scope consistent with the principles and features described herein.

Figure 1 is a block diagram of a conventional computer system. As is seen, the computer system 10 includes a central processing unit (CPU) 12. In this embodiment, the CPU is coupled to a bridge 14. Bridge 14 is coupled to a bus 16 and a system memory 18. The bus 16 in a preferred embodiment is an AGP or PCI bus, which allows for signals to be sent thereon to I/O device 20, I/O device 22, graphics controller 24, and frame buffer 26. As is seen, the graphics controller 24 controls a frame buffer 26 and also sends video information (generally RGB) via a digital analog converter (DAC) 27 to a monitor 28.

It should be understood that the elements of the computer system 10 are shown as discrete components, they can be combined in a variety of fashions and still be within the spirit and scope of the present invention. For example, the bus 16 could be merged into the system, so that there is no actual system bus between the various components but in fact the various components communicate directly. In addition, the graphics controller 24, CPU 12, frame buffer 26, system memory 18 and I/O devices 20 and 22 could be integrated in any fashion and that combination would be within the spirit and scope of the present invention.

The following discussion will describe two conventional techniques for rendering font characters into a memory of the computer system 10. These conventional techniques will be described in the context of pixels being rendered into the frame buffer 26. However, one of ordinary skill in the art readily recognizes that the pixels of information could be rendered into any memory device within the computer system 10. For example the system memory could be utilized in this fashion.

In a first conventional system, the frame buffer 26 receives bytes of information directly from the CPU 12 to render a font characters in the frame buffer 26. Figure 2 illustrates the rendering pixels of information directly by the CPU 12 to the frame buffer 26. Typically, each of pixel are at least a byte in size, although one of ordinary skill in the art recognizes the pixels could be other sizes such as 16 bits or 32 bits. As is well known, the frame buffer 26 is rendered out typically from left to right. In this embodiment, as seen, the font character 100 is the letter "T". As is also seen in this embodiment if each pixel represents a byte of information, the letter is 5 bytes wide by 7 bytes high. The hexadecimal number 00 106 represents one byte of one color. The hexadecimal number FF 104 represents one byte of another color. Typically, the DAC 27 receives the information and can, for example, convert the 8 bit number to a 24 bit (8 bits each for

the red, green and blue channels) value. It should be understood by one of ordinary skill in the art that the color could be represented by any numbers and is not to be limited to the 00 and FF hexadecimal numbers or to one byte per pixel described herein.

As is seen, if it is assumed that each pixel of the font represents a byte of information, then 35 bytes of information are required to represent the particular font character. Therefore, if the frame buffer is an 8 bit frame buffer, the CPU would be required to write 35 bytes directly to the frame buffer to render the font. If the frame buffer is a 16 bit frame buffer, then the CPU would be required to write 70 bytes to render the font. If frame buffer is a 32 bit frame buffer, then it would require 140 bytes for the CPU to render the font. Accordingly, as is seen, as more and larger font character need to be rendered into the frame buffer 26 the processing power and processing time increases significantly.

A second conventional technique rendering characters into a memory reduces processing power and time by providing lossless compression. This is particularly advantageous when rendering font characters into memory because typically fonts are two colors. Therefore, since there are only two colors (for example, black and white), then it is possible to compress the image by having a bit represent a pixel on the frame buffer. It is also possible to provide a transparent mode to further reduce the processing power and time that would require that only one of the colors to be rendered in memory.

Sub A2

Figure 3 illustrates a glyph or a compressed font 300 produced in accordance with the second conventional technique. Pixels are represented by a bit of information (i.e., a 1 or an 0) As is seen, each bit 302 represents one of the bytes of the information that was shown previously in Figure 2. Accordingly, rather than having to provide 35 bytes of information in the case of an 8 bit frame buffer, in this environment the CPU 12 (Figure 1) would only need to provide 35 bits

across the bus 16, which translates into a 4 3/8 bytes of information. These 4 3/8 bytes of information therefore can be decompressed utilizing circuitry in the graphics controller 24.

Figure 4 illustrates a graphics controller 24 and a frame buffer 26 in accordance with the second conventional technique. Typically, the graphics controller 24 will include an expansion block 302 which will render the pixel into the frame buffer 26 based upon the glyphs. Th expansion block 302 includes first and second registers 312 and 314. A multiplexer 310 within the expansion block 302 is utilized to select between the two registers 312 and 314 based upon the glyph. The selection of the registers 312 and 314 is based upon the bits provided by the CPU 12 (Figure 1). Accordingly, if the bit provided by the CPU 12 is a foreground color, the multiplexer 310 selects the register 312 which would expand the bit to a plurality of bits (i.e., 8 bits, 16 bits or 32 bits) and those bits are then provided into the frame buffer 26 as a foreground color. On the other hand, if the bit provided is background color, the multiplexer 310 selects register 314 which would expand the bit into a plurality of bits and those bits are then provided to the frame buffer 26 as a background color. As further improvement in the second embodiment, a so-called transparent mode can be provided. In this mode, a multiplexer 311 allows for the background color to be provided from the frame buffer 26 via select line 315 and input 313. For example, if the background color is 00 then the frame buffer automatically loads zeros for the background color of that particular character. In so doing, rendering time is further reduced. Accordingly the character 100 that is rendered in the frame buffer is exactly the same as that shown in Figure 1.

Through the use of a glyph, it is possible to render information more efficiently than if the CPU 12 is directly writing the information to a memory. Accordingly, for an 8 bit frame buffer there is a 8:1 compression ratio, for a 16 bit frame buffer is a 16:1 ratio or for a 32 bit frame buffer, there is 32:1 compression.

However, the problem with the second conventional technique is that it requires that the CPU 12 input a variety of information that requires several transfers from the CPU 12 to the graphics processor 24. Accordingly, the CPU must first find the font, find the character in the font, and figure out how many bytes to transfer and transfer all the bytes.

To more fully explain this problem, refer now to the following. To reiterate, the graphics controller 24 includes an expansion block 302. The expansion block 302 in a preferred embodiment includes first and second registers 312 and 314. These registers 312 and 314 provide for the foreground and background colors respectively. The foreground and backgrounds are selected by the multiplexer 310 to provide the particular character to the frame buffer 26.

As is seen, a register set 301 is within the graphics controller 24. The register set 301 includes the x-coordinate register 304, a y-coordinate 306, a size width register 308 and a size height register 309 of the font character. In operation, the user of the CPU 12 loads the x and y coordinates registers to provide position information for the font character. The CPU 12 also loads the size height and the size width registers 308 and 309 to provide size information for each font character that is to be rendered into the frame buffer 26. Thereafter each character must be written across the bus to the graphics controller 24. Accordingly, a significant number of transfers are required to ensure that the graphics controller 24 provides the information efficiently to the frame buffer 26. What is desired is to be able to transfer this information more efficiently while still operating in the environment of a conventional computer system.

Accordingly, a system and method in accordance with the present invention provides for a plurality of font arrays to be provided within the memory of a computer system. The memory could be the frame buffer 26, system memory 16 or any other memory within the computer system 10. The graphics controller includes a mechanism which allows for a font array in the memory to

be accessed by the graphics controller. The graphics controller also includes a mechanism for allowing each font character to be rendered into the memory. In so doing the number of transfers from the CPU is significantly reduced.

Sub A3

To discuss the present invention in the context of a preferred embodiment refer now to Figure 5 and the accompanying discussion. Figure 5 is a block diagram of a graphics controller 400 and a frame buffer 450 in accordance with the present invention. The graphics controller includes an expansion unit 402 that has similar components to that of expansion unit 302 of Figure 4. The memory 450 includes a data structure 451. The data structure 451 includes a plurality of font arrays 460 and 462. It should be understood that there could be any number of font arrays in the data structure 451. As is seen, in font array 462 the size of the font characters are larger than the size of the font characters in font array 460. Hence, for each font array 460 and 462 a different index and a different font pointer is required to allow the graphics controller to access the associated font characters. Accordingly, a font pointer is changed when, for example, the font is changed. Each of the font arrays 460 and 462 include information concerning the size of each character and an index that indicates the location of each font character within the font array. The font characters can be either full sized fonts or glyphs. The information within the font array is utilized by the graphics controller 400 to allow the graphics controller 400 to render a particular character retrieved from the memory 450.

The graphics controller 400 includes a plurality of registers to utilize the information related to the font arrays 460 and 462 such that characters can be efficiently retrieved from the frame buffer 450. The various registers and their function are described below.

INDEX Register 412

This register contains the index to the character in the font array.

FONT Pointer Register 414

This register specifies the location of the font array and bytes allocated to each character. The register 414 includes an offset field and a pitch field.

FONT Pitch Register 415

This register specifies the number of bytes between the beginning of each character.

X Register 416

This register field contains, for example, the horizontal location of the output monochrome rectangle's upper left corner.

Y Register 418

This register contains, for example, the vertical location of the output monochrome rectangle's upper left corner.

~~**SIZE Width Register 42**~~

Sw3 A4

This register contains the width of the output monochrome rectangle. If width is zero, nothing is rendered.

SIZE Height Register 421

This register contains the height of the output monochrome rectangle. If height is zero, nothing is rendered.

The graphics controller 400 also includes a register 410 that provides the glyph information of a particular character to an expansion block 402 of the graphics controller 400. As is seen, in this embodiment, the particular character that is to be rendered into the memory 450 is the letter "T".

OPERATION

In a preferred embodiment the user loads the font pointer register 414 with a value that points to a font array at an address 470 or to a font array at an address 471 of the memory 450. The user then loads values into the index register 412, x or horizontal information register 416 and y or vertical information register 418. One of ordinary skill in the art recognizes either the x or y registers could also be linear information registers. The graphics controller 400 then looks up the font in the memory 450. The graphics controller 400 then locates the character within the font array by the index from the index register 412. For example, in font array 460 the character 426 would be at index 0. The size width and size height information is part of the character or glyph information. The graphics controller 400 reads that character information from the memory 450 into the register 410 which holds the glyph information. Therefore, the CPU never has to load the width and height information into the registers. This glyph information is then provided to the expansion unit 402. The expansion unit 402 will render the information to the memory portion 450 at the appropriate locations. The x and y coordinates retrieved from registers 416 and 418 would be used to locate within the memory 450 where the particular character was to be written.

The size width register 420 provides information concerning the size of the particular character to the memory 450. The size height register 421 provides information concerning the height of the particular character to the memory 450.

Accordingly, in a system and method in accordance with the present invention, the graphics controller performs most of the transfers to the frame buffers and the CPU only needs to transfer minimal data. Therefore, by providing font arrays in the memory and allowing the graphics processor to access those font arrays, the graphics processor can then obtain the font character as a glyph and then render the character on the memory.

545 AS

In an example, all the user needs is to load the index value, the x value and y value for the character that is to be, where the index is an ASCII character number. In a preferred embodiment, the x value can be 12 bits, the y value can be 12 bites and the index value can be 8 bits. In so doing, one 32 bit transfer from the CPU to the graphics controller is all that is needed to render the character.

Although the present invention has been described in accordance with the embodiments shown, one of ordinary skill in the art will readily recognize that there could be variations to the embodiments and those variations would be within the spirit and scope of the present invention. For example, one of ordinary skill in the art readily recognizes that the pixels of information could be retrieved from and rendered into any memory device within the computer system. In addition, in the system and method in accordance with the present invention the graphics controller 400 could retrieve characters from a font array that is located in the system memory and render the retrieved characters into the frame buffer 450. Accordingly, many modifications may be made by one or ordinary skill in the art without departing from the spirit and scope of the appended claims.